Quelques principes fondamentaux de Git

Pierre-Olivier Parisé

21 octobre 2019

Git est un logiciel puissant de versionnement de projets. Pour un informaticien, il s'agit d'un outil maintenant indispensable pour la bonne gestion de ses projets informatiques, autant professionnels que personnels. Dans ce court document, je vous présenterai quelques principes/-concepts de base à connaître à tout pris pour bien comprendre le fonctionnement de Git. Ces principes sont les « commits », le fichier « .log » et les branches.

1 Qu'est-ce qu'un commit ?

Un « commit » est une façon d'indiquer à Git que vous êtes prêt à apporter des modifications à un/des fichier.s de votre projet. Afin de le spécifier à Git, il faut entrer, en ligne de commandes, la commande `git add nomfichier.ext` pour chacun des fichiers et ensuite `git commit -m 'Message explicatif'`.

La première commande permet d'indiquer à Git quels sont les fichiers qui sont ajoutés au « commit ». La deuxième commande est le « commit » en tant que tel. La partie à partir du `-m` permet d'écrire un message qui explique les motifications apportées aux fichiers.

Un « commit » peut être vu comme l'étape précédent l'envoi d'un colis. Une fois toutes les informations nécessaires pour l'envoi du colis sont rassemblées, celui-ci est placé sur les étagères de la compagnie de transport, attendant son envoi ; il est prêt à être envoyé à destination. Cette dernière étape est le « push » du colis à destination. Avec le logiciel Git, cette action s'effectue à l'aide de la commande `git push origin master`.

2 À quoi sert la commande git log

Reprenons l'exemple de l'envoi du colis. Nous avons maintenant mis notre colis sur l'étagère des colis prêts à être envoyés. D'autres colis seront déposés sur cet étagère et « commités » à différentes destinations. Il est important d'avoir une sorte d'historique de chaque colis placé sur l'étagère. Une sorte de fichier « log ».

La commande `git log` permet justement d'accéder à cette historique. Dans le contexte du programme Git, nous accédons à l'historique des « commits » reliés au projet et des « push » du projet via la commande `git log`.

3 Qu'est-ce qu'une branche ?

Poursuivons avec l'exemple du colis. Imaginons que le colis doit se rendre de la ville de Québec jusqu'à Paris. Le trajet principal est que le colis soit envoyé dans un vol direct Québec-Paris. Cependant, il arrive que le colis doit passer par des vols intermédiaires afin de se rendre à destination.

On peut dire que le trajet principal est la branche principal du projet et que l'idéal serait de pousser nos modifications sur cette branche principal. Or, ceci n'est pas toujours le cas, surtout si on souhaite modifier une certaine fonctionnalité du projet, sans affecter son entièreté. On peut appeler les vols intermédiaires comme des branches secondaires sur lesquels on travaille. Pour créer une nouvelles branche dans Git, on tape la commande suivante : git branch nomBranche. Ensuite, afin de se positionner sur la branche [1], on tape en ligne de commande git checkout nomBranche. Enfin, lorsque nous voulons joindre nos modifications à la branche principale, on doit

1. se positionner sur la branche principale avec la commande git checkout master ;
2. taper la commande git merge nomBranche.

En joingnant la branche, il se peut que des conflits se produisent. On peut les résoudre directement en ouvrant les fichiers.

1. Ceci permet de travailler comme d'habitude sur une autre branche.